W krajach zachodnich mikrokomputery nie są w zasadzie wykorzystywane do... wykonywania obliczeń. O karierze komputera osobistego przesądziły zastosowania w dziedzinach dalekich od wyliczania całek, różniczek itp. Procesor tekstowy czy też programy „biurowe" nie wychodzą z reguły poza podstawowe cztery działania. Dlatego też w podawanych parametrach mikrokomputera niezmiernie rzadko spotyka się dokładność numeryczną obliczeń.

Z kontaktów z czytelnikami wynika, że krajowi użytkownicy mikrokomputerów dzielą się na fanatyków rozrywki (gry komputerowe) i sfrustrowanych naukowców, którzy — zniechęceni sposobem funkcjonowania naszych ośrodków obliczeniowych — pragną poprawić sobie nieco warunki pracy, choćby kosztem spożywania suchego chleba na stypendiach zagranicznych. Dlatego też opracowanie dobrych testów numerycznych wydaje się być — z naszej polskiej perspektywy — sprawą dość istotną. Można by wprawdzie zalecić korzystanie ze sprzętu firm specjalizujących się w obsłudze naukowców (np. HEWLETT PACKARD), ale biorąc pod uwagę jego ceny — wielu jajogłowych przypłaciłoby to zagłodzeniem. Zapewne los głodujących na Zachodzie naukowców (naszych) wzruszy Czytelników, oczekujemy więc na propozycje ciekawych testów dokładności numerycznej.

Jaka dokładność numeryczna

Zamieszczone w mikroKLANIE 3 programy testujące nie pozwalały na porównanie dokłaności numerycznej komputerów. Jednak w wielu zastosowaniach właśnie dokładność numeryczna, a nie czas obliczeń, odgrywa zasadniczą rolę. Poniżej prezentujemy tekst dokładności obliczeniowej (pierwiastek kwadratowy, kwadrat liczby oraz generator liczb losowych) i jego wyniki dla popularnych komputerów osobistych. Jak poprzednio, zostawiamy wolne miejsce w tabeli na testy innych komputerów, posiadanych przez Czytelników. A może ktoś zaproponuje uniwersalny test wszystkich funkcji numerycznych dla języka BASIC?

```
10 R=0
20 S=0
30 FOR N=1 TO 100
40 A=N
50 FOR I=1 TO 10
60 A=SQR (A)
70 R=R+RND (1)
80 NEXT I
90 FOR I=1 TO 10
100 A=A**2
110 R=R+RND (1)
120 NEXT I
130 S=S+A
140 NEXT N
150 PRINT ABS (1010—S/5)
160 PRINT ABS (1000-R)
170 END
```

Porównanie parametrów dokładności numerycznej najbardziej popularnych modeli komputerów osobistych

Komputer	Czas[1])	Dokładność	Losowość
BBC Acorn	0 : 21	0,0000128746033	5,2
IBM PC	0 : 24	0,01159668	6,3
TRS 80/4	0 : 53	0,0670776	6,5
HP 85A	1 : 20	0,00000002	5,8
HP 75C	1 : 38	0,00000002	5,8
VIC 20	1 : 49	0,0010414235	23,7
CBM 64	1 : 53	0,0010414235	8,9
Apple IIe	1 : 53	0,0010414235	12,0
TI 99/4A	3 : 46	0,00000011	2,6
ZX81	4 : 23	0,0006685257	6,3
SPEC-TRUM	4 ; 39	0,0006685257	3,5
HP 41C	31 ; 48	0,000034000	7,8[2])
,,,,,,,,,,,,,	,,,,,,,,,	,,,,,,,,,,,,,,,,,,,,	,,,,,,,,,

1) minuty : sekundy
2) generator liczb losowych z PPC ROM, funkcja RN

Wyniki testów: przypisujemy wartości 0,0000001 dokładność bardzo dobrą, wartościom powyżej 0,01 — złą; podobnie losowość poniżej 10 jest dobra, powyżej 10 — zła.

Opracował
JAKUB TATARKIEWICZ
na podstawie „Creative Computing", styczeń 1984

Strach przed ?

Amerykańskie pismo POPULAR COMPUTING, zajmujące się mikroinformatyką, opublikowało niedawno (nr 6/84) artykuł Jerry Pournelle'a o perspektywach tej dziedziny w różnych krajach. Autor twierdzi kategorycznie, że w państwach RWPG, a w tym także w PRL, mikroinformatyka nie będzie rozwijana. Wprost przeciwnie — będzie świadomie hamowana, jako zagrażająca kontroli przepływu informacji. Podobny pogląd wyraził — w ostrej propagandowo audycji — GŁOS AMERYKI.

To groźne. Bo choć nie widać poważnych argumentów na uzasadnienie takiej tezy — do powstania w Polsce niezależnej sieci informacyjnej wykorzystującej mikrokomputery jest naprawdę bardzo daleko — to jednak znajdują się one post factum. Urzędnik przestaje pytać, do czego służy komputer, a zaczyna kojarzyć nożyce ze stołem.

Do Przemysława Filipowicza z Lubartowa nadeszła w kwietniu 1984 br. przesyłka od brata z Toronto (Kanada), zawierająca mikrokomputer COMMODORE VIC-20, zasilacz, magnetofon kasetowy, komplet kaset do gier telewizyjnych i nauki języka angielskiego oraz przewody połączeniowe. Urząd Celny w Gdyni odmówił wydania pozwolenia na przywóz, stwierdzając, że:

„Urząd Celny mógłby wydać pozwolenie na przywóz przedmiotowego komputera tylko w przypadku, gdyby komputer ten był niezbędny do pracy zawodowej względnie naukowej Obywatela".

Udowodnij, żeś nie nosorożec! Obywatel wysłał zaświadczenie o wykonywanych przez siebie usługach w dziedzinie elektromechaniki samochodowej i o żonie — ekonomistce, opisał niedouczenie politechniczne dzieci (11 i 17 lat). I nic... Nosorożec! Gdynia komputera nie przepuściła.

Skończyło się jednak pozytywnie. W sierpniu (!) Główny Urząd Ceł w Warszawie uwzględnił odwołanie. Okazało się przy tym, że wprawdzie nie ma żadnych formalnych ograniczeń w przywozie komputerów (poza koniecznością uzyskania pozwolenia...), ale niedawno (ponoć w czerwcu) trochę się „zaostrzyło". Przygotowywana jest ponadto — jak powiedziano w GUC — normalizacja prawna tego rodzaju przywozów.

Niby chcemy gonić świat, nie pozostawać w zaścianku... Ale jakże tu nie dostrzegać tak rozgłaszanych zagrożeń. Więc na wszelki wypadek wzmożemy czujność. Bo — jak stwierdził kompetentny urzędnik GUC — decyduje przecież dobro państwa.

ZBIGNIEW GLUZA

- Na rynku brytyjskim pojawiły się pierwsze komputery osobiste w standardzie MSX (zaproponowanym przez MICROSOFT). Ceny w granicach 250—300 funtów. Jest to ponad dwukrotnie więcej niż w Japonii, kraju producentów! Szkoda, rozwiązania techniczne w standardzie są bardzo ciekawe — ale ta cena...

- Witold Majewski (Wrocław tel. d, 55-48-96; służb. 67-90-71 w. 203) jest posiadaczem ZX SPECTRUM i bogatego oprogramowania (m.in. ASSEMBLER, FORTH, VU- FILE, High Resolution), które chętnie udostępni innym w drodze wymiany. Poszukuje również papieru termicznego (szer. 10 cm) do drukarki.

Opisywaliśmy już sposób zrealizowania współpracy z magnetofonem kasetowym. Obecnie zamierzamy przedstawić rozwiązanie umożliwiające przyłączenie do mikrokomputera napędu dysków elastycznych. Pamięć masowa tego typu jest niewątpliwie wielokrotnie droższa, umożliwia jednak zainstalowanie jednego z popularnych systemów operacyjnych (np. CP/M) i wykorzystanie stworzonego oprogramowania do nieco poważniejszych zastosowań niż gry i zabawy.

Zanim opiszemy konkretną realizację sprzęgu dla mniej otrzaskanych Czytelników przygotowaliśmy tekst wprowadzający w zagadnienia związane z dyskami elastycznymi.

Elastyczne dyskietki

Nie istnieje komputer bez pamięci. W erze komputerów-mamutów towarzyszyły im równie wielkie (rozmiarami) i równie powolne pamięci. Nie będziemy w tym miejscu robić przeglądu historycznego — dość powiedzieć, że w ramach poszukiwań rozwiązania, które pozwoliłoby na tanie przechowywanie dużych ilości łatwo dostępnej informacji, wymyślono elastyczne dyskietki. Od momentu wprowadzenia tej techniki upłynęło już trochę czasu i komputery zdążyły na tyle stanieć, że to co kiedyś wydawało się tanie, dzisiaj wymaga często większych nakładów finansowych niż sam komputer. Mimo dalej prowadzonych na szeroką skalę poszukiwań (EAROM-y, pamięci domenowe) na razie nie ma rozwiązania, które za niższą cenę oferowałoby równie „elastyczny" sposób przechowywania informacji.

Dyskietka to krążek sztucznego tworzywa (mylar), pokrytego tlenkiem metalu (nośnik magnetyczny) i zapakowanego w kartonową kopertę chroniącą przed pyłem i uszkodzeniami.

Opracowano wiele standardów dotyczących sposobu zapisywania informacji. Dla dyskietek o średnicy 8" wykorzystuje się zazwyczaj 77 ścieżek, dzielonych na sektory. Istnieją dwie metody wydzielania sektorów na ścieżce. Jedna z nich — mechaniczna (ang. hard-sectors) — polega na wyznaczaniu sektorów przez specjalnie wykonane w dyskietce otwory. W drugiej — na ścieżce zapisuje się specjalną informację wydzielającą poszczególne sektory (ang. soft-sectors). Metoda mechaniczna pozwala na zapis większej ilości informacji, narzuca jednak określoną długość sektora. Druga metoda, choć bardziej rozrzutna, zdobyła większą popularność, gdyż wykorzystywana jest przez potentata informatycznego — IBM.

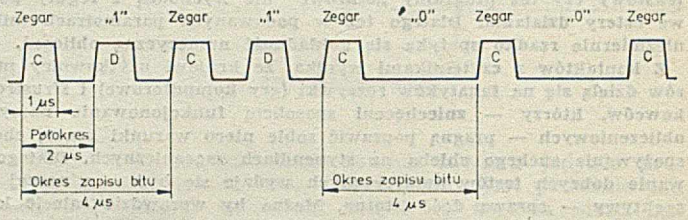
Najczęściej wykorzystywane są dwa standardowe formaty zapisywania danych:
— IBM 3740, tzw. pojedyncza gęstość (SD, ang. Single Density)
— IBM 34, tzw. podwójna gęstość (DD, ang. Double Density).

W zależności od konstrukcji używanego napędu dyskietek, informacja może być zapisywana tylko z jednej strony dyskietki (SS, ang. Single Side) lub z obydwu (DS, ang. Double Side) — w tym drugim przypadku konieczne są dwie głowice zapisująco-odczytujące.

Kodowaniej dla pojedynczej gęstości

Format IBM 3740 wykorzystuje technikę modulacji częstotliwości (FM, ang. Frequency Modulation) dla zakodowania strumienia danych zapisywanego na ścieżce. W rozwiązaniu tym bity danych rozdzielane są impulsami zegarowymi, co zapewnia utrzymanie stałej synchronizacji w trakcie odczytu. Rzut oka na rysunek 1 wystarczy do zrozumienia, dlaczego ten sposób kodowania nazywany jest modulacją częstotliwości.

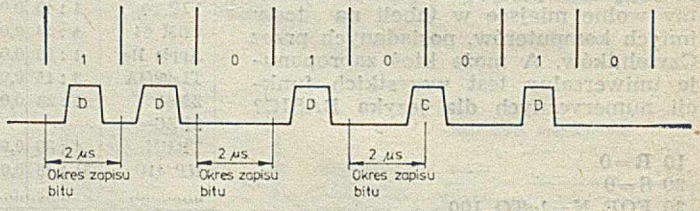
Przy odczycie informacji należy nie tylko przekształcić postać danych z szeregowej na równoległą, ale także odseparować zegar. Istnieją dwie metody separacji. Pierwsza polega na zastosowaniu generatora monostabilnego. Mimo niewielkich kosztów i prostej realizacji, nie zaleca się jej stosowania ze względu na małą odporność na zakłócenia wynikające z nierównomiernej pracy zespołów mechanicznych. Drugi sposób wymaga zastosowania układu generatora z zamkniętą pętlą fazową (PLO, ang. Phase Locked Oscilator). W przypadku kodowania z podwójną gęstością konieczne jest zastosowanie układu PLO.



Rys. 1. Sposób kodowania danych dla zapisu o pojedynczej gęstości — modulacja częstotliwości (FM)

Kodowanie dla podwójnej gęstości

Zwiększenie ilości informacji przechowywanej na dyskietce przez zagęszczenie zapisu nie jest praktykowane, gdyż stosowana gęstość bliska jest fizycznemu ograniczeniu dla stosowanego nośnika magnetycznego. Można jednak zastosować „gęstszą" metodę kodowania. Standard IBM 34 jest zmodyfikowaną metodą modulacji częstotliwości (MFM. ang. Modified Frequency Modulation). Ograniczono w nim występowanie impulsów zegarowych do sytuacji, gdy zachodzi konieczność rozdzielenia bitów zerowych. O ile bitowi „1" odpowiada zapisanie impulsu, o tyle dla bitu „0" impuls nie jest zapisywany. Dla rozgraniczenia ciągu bitów zerowych trzeba więc zapisać impulsy zegarowe. Sposób kodowania MFM został zobrazowany na rysunku 2. Ponieważ dla zmodyfikowanej modulacji częstotliwości liczba impulsów zmniejsza się o połowę, okres trwania komórki informacji można zredukować z 4 µs do 2 µs. Minimalny odstęp między impulsami w obydwu standardach wynosi 2 µs.



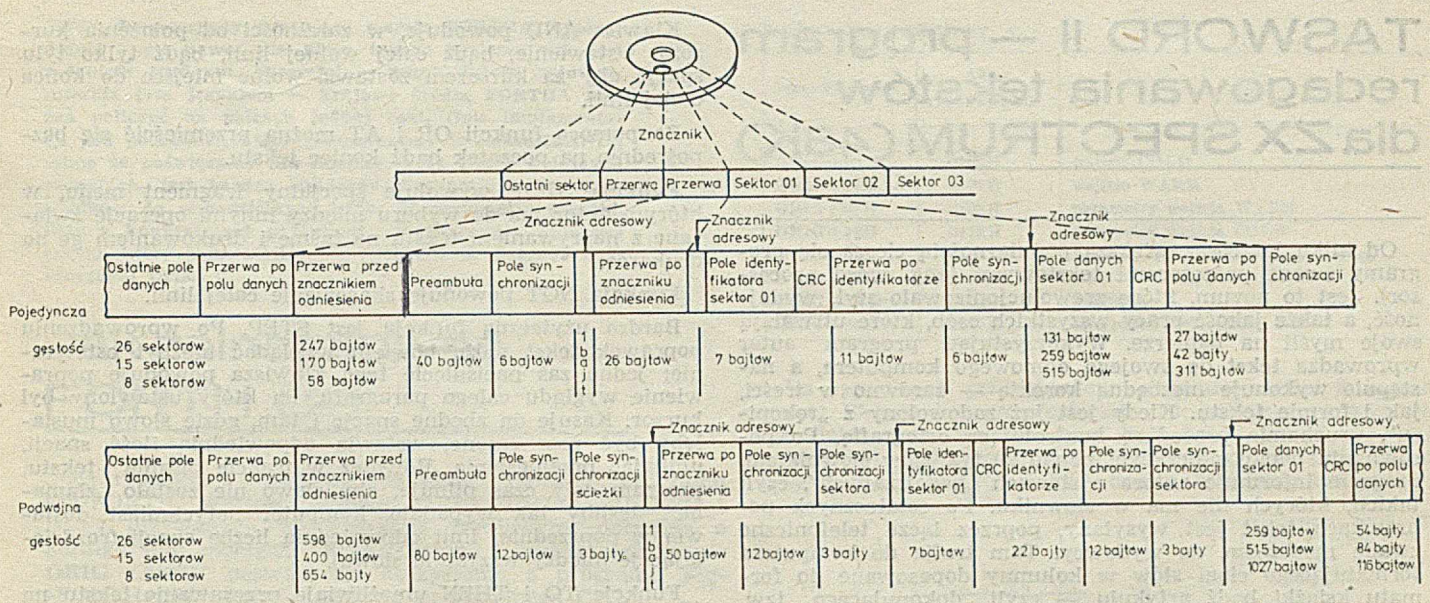
Rys. 2. Sposób kodowania danych dla zapisu o podwójnej gęstości — zmodyfikowana modulacja częstotliwości (MFM)

Organizacja zapisu informacji na ścieżce

Każda ścieżka może zostać podzielona na 26, 15 lub 8 sektorów. Podział jest dokonywany na etapie inicjowania dysku, czyli tzw. formatowania. Dyskietki są zazwyczaj sformatowane przez producenta, jednak większość systemów wspomagania projektanta oprogramowania również może realizować to zadanie. W trakcie formatowania informacja zapisana wcześniej na dyskietce zostaje skasowana.

W zależności od zastosowanego podziału ścieżki, sektor może zawierać 128, 256 lub 512 bajtów dla SD i 256, 512 lub 1024 dla DD.

Na rysunku 3 przedstawiono organizację zapisu na ścieżce. Punktem odniesienia jest fizyczny znacznik początku (otwór). Pierwszy sektor występujący za znacznikiem ma zawsze przypisany numer 1. Następne sektory mogą być ponumerowane w dowolnej kolejności. Numer sektora zapisywany jest w polu identyfikującym (ang. ID field) w trakcie formatowania dyskietki. Pola identyfikacji i pola danych są rozgraniczane przerwami (ang. gap).



Rys. 3. Struktura zapisu na ścieżce dyskietki dla zapisu o pojedynczej i podwójnej gęstości

Każde pole rozpoczyna się od sześciu bajtów składających się z samych zer (ang. sync field), po których następuje 1-bajtowy znacznik adresu (ang. address mark). Stosowane są cztery typy znaczników adresu (o różnych kodach) dla oznaczenia:

— fizycznego początku ścieżki
— początku pola identyfikującego
— początku pola danych
— początku pola skasowanych danych.

Pole identyfikujące składa się z siedmiu bajtów, w których kolejno zapisane są: znacznik adresu, numer ścieżki, numer głowicy, numer sektora, rozmiar sektora oraz dwa bajty CRC (ang. Cyclic Redundancy Check).

Pole danych składa się ze znacznika adresu, odpowiedniej liczby bajtów danych i dwóch bajtów CRC.

Sprzęg z napędem dyskietek elastycznych

Z chwilą włożenia dyskietki do kieszeni napędu i zamknięcia osłonki kończy się obsługa manualna. Dalsza współpraca z napędem realizowana jest poprzez sterownik sprzęgu.

Zasadniczym zadaniem jest ustawienie głowicy zapisująco-odczytującej nad żądaną ścieżką. Jest to realizowane przez silnik krokowy i dlatego logika sprzęgu dostarcza sygnał w rodzaju: krok w przód, krok w tył. Jeżeli rozpoczynamy współpracę z dyskietką, stosowany jest specjalny mechanizm (dioda i fotokomórka), pozwalający odnaleźć punkt odniesienia, czyli — ścieżkę zerową. Głowica ustawia się automatycznie na ścieżkę zerową — w wyniku zerowania lub wykrycia błędu pozycjonowania.

Do odnalezienia konkretnej ścieżki konieczna jest znajomość aktualnej pozycji głowicy. W tym celu sterownik napędu zazwyczaj zawiera specjalny rejestr, w którym przechowywany jest numer ścieżki, nad którą znajduje się głowica. Dzięki temu możliwe jest wygenerowanie odpowiedniej liczby kroków w przód lub do tyłu. Prawidłowość ustawienia głowicy weryfikowana jest przez porównanie z informacją zapisaną w polu identyfikującym.

Dla dokonania operacji zapisu lub odczytu niezbędne jest opuszczenie głowicy na powierzchnię dysku. Jest to tzw. ładowanie głowicy. Przed przesunięciem na inną ścieżkę głowica nie musi być podnoszona.

W sprzęgu z napędem dyskietek stosowane są również sygnały nie związane bezpośrednio z obsługą funkcji mechanicznych. Linia SELECT wykorzystywana jest do wybrania napędu lub głowicy, za pomocą której realizowana będzie operacja zapisu lub odczytu. Z kolei linia READY (sterowana przez układy elektroniczne napędu) sygnalizuje możliwość współpracy (włączone zasilanie, włożona dyskietka, zamknięta osłona itp.).

Niektóre napędy mają dodatkowo możliwość sygnalizowania, że dyskietka jest zabezpieczona przed zapisem (ang. Write Protect) oraz sygnalizację uszkodzeń układu zapisu (ang. Write Fault).

W miarę przesuwania się w kierunku osi dysku długość ścieżek maleje, co powduje większą gęstość zapisu.

Dla skompensowania tego zjawiska, od połowy dyskietki stosuje się mniejszy prąd zapisu. Linia sterująca tą funkcją nazywana bywa „LOW CURRENT" lub „TG43" (ang. track greater than 43).

W następnym numerze mikroKLANU opiszemy przykład konkretnej realizacji sprzęgu.

AJP

- Od 1 października w klubie Wydawnictwa SIGMA (Warszawa, ul. Mazowiecka 12) obok zwyczajowego spotkania hobbystów-mikroinformatyków prowadzona będzie giełda sprzętu i oprogramowania. Ponieważ na „Perskim" zabroniono tego typu działalności, organizatorzy pragną w ten sposób podtrzymać tak cenne dla hobbystów forum wymiany. Planowane jest organizowanie giełd co miesiąc. Zapraszamy wszystkich zainteresowanych, a szczególnie tych, którzy mają coś do zaoferowania innym.

- Użytkownicy mikrokomputerów MERITUM I oraz ZX-81 poszukują kontaktów z innymi posiadaczami tego sprzętu lub TRS 80/II. Szczególnie interesuje ich oprogramowanie — tak w BASICU, jak i w języku maszynowym Z80. Na razie korzystają z czasopisma BYTE; będą zobowiązani za udostępnienie innych tytułów z gotowymi programami. Kontakt: Politechnika Łódzka, centr. tel. 36-55-22; Zbigniew Chrześcijański — tel. w. 673, Jacek Żebrowski — tel. w. 674; ew. wiadomość na tel. bezp. w Łodzi — 36-76-44.

TASWORD II — program redagowania tekstów dla ZX SPECTRUM (48K)

Od kilku lat coraz większą popularnością cieszą się programy zwane procesorami tekstowymi (ang. word processor). Jest to novum, które zrewolucjonizowało styl, wydajność, a także jakość pracy wszystkich osób, które utrwalają swoje myśli na papierze. Wykorzystując program autor wprowadza tekst do swojego domowego komputera, a następnie wykonuje niezbędną korektę — zarówno w treści, jak i formie tekstu. Kiedy jest już zadowolony z „rękopisu", uruchamia procedurę kontrolującą ortografię. Po porównaniu tekstu z liczącym ponad 20 tys. słów słownikiem, program informuje autora o słowach „podejrzanych", czyli takich, których nie ma w słowniku. Po ostatecznych poprawkach tekst jest wysyłany, poprzez łącze telefoniczne zwane modemem, do wydawcy. Tam trafia do komputera formatującego ciągi słów w kolumny dopasowane do formatu książki bądź artykułu — czyli dokonującego tzw. składu. Po czym, zależnie już tylko od wydajności drukarni, można rozpocząć kolportaż wydrukowanego tekstu, ukończonego przez autora kilka dni wcześniej.

Najciekawsza cecha programu TASWORD II dla ZX SPECTRUM jest to , ze na jednej linii potrafi on zmiescic 64 znaki. Kazdy kto pracowal z ZX SPECTRUM na pewno narzekal na mala (32) liczbe znakow w linii. Omawiany program, wykorzystujac specjalne procedury napisane w jezyku maszynowym, drukuje znaki o szerokosci czterech punktow. Wbrew pozorom sa one calkiem czytelne. 1]

Gdyby jednak ktoś upierał się, że litery są za mało czytelne, zawsze może „włączyć" opcję **c**, która pokazuje na ekranie aktualnie poprawiany bądź wpisywany wycinek tekstu, przy wykorzystaniu standardowej wielkości liter.

Aby autorowi opłacało się wykorzystywać omawiany program, musi on w istotny sposób ułatwiać prace. Służy temu cała gama specjalnych funkcji.

Jedną z najbardziej użytecznych jest EDIT. Po wprowadzeniu jej na ekranie pojawia się strona pomocnicza, na której w wielkim skrócie opisane są wszystkie możliwe polecenia (tzw. menu). Po naciśnięciu obydwu klawiszy SHIFT, wyświetlane są te polecenia, których zastosowanie wymaga równoczesnego naciśnięcia ich obu.

Klawisze 3 i 4, czyli TRUE VIDEO i INV. VIDEO, służą do szybkiego przenoszenia kursora przez tekst. Szybkość jest uzyskiwana dzięki temu, że kursor zatrzymuje się tylko na początku każdego słowa.

Klawisze 5, 6, 7 i 8 służą do przesuwania kursora w dowolne miejsce ekranu.

Klawiszem 9 uruchamia się opcje wstawiania w tekst specjalnych znaków sterujących przeznaczonych dla drukarki. Znaczenie tych znaków może zaprogramować sam użytkownik.

Klawisz <> powoduje równomierne rozrzucanie tekstu w linii, w której znajduje się kursor.

Klawisz AND powoduje, w zależności od położenia kursora, wstawienie, bądź całej wolnej linii, bądź tylko tylu spacji, aby za kursorem zostawić wolne miejsce do końca danej linii.

Za pomocą funkcji OR i AT można przemieścić się bezpośrednio na początek bądź koniec tekstu.

Funkcja STOP wywołuje specjalny fragment menu, w którym autor ma do wyboru między innymi operacje związane z nagrywaniem tekstu na taśmę i drukowaniem go na drukarce.

Funkcja NOT powoduje skasowanie całej linii.

Bardzo użyteczną funkcją jest STEP. Po wprowadzeniu poprawek tekst może bowiem wyglądać niezbyt estetycznie; jedno zaś naciśnięcie tego klawisza powoduje poprawienie wyglądu całego paragrafu, na który ustawiony był kursor. Kasuje on zbędne spacje i tam, gdzie słowo musiałoby być przenoszone, wstawia odpowiednią ilość spacji, unikając przeniesienia. Również w trakcie pisania tekstu, program cały czas pilnuje, aby słowo nie zostało „złamane". Jeśliby tak wypadało, komputer natychmiast dostawia w poprzedniej linii odpowiednią liczbę spacji (rozdzielając je między wszystkie słowa).

Funkcje TO i THEN umożliwiają przesuwanie tekstu na ekranie w górę lub w dół.

Klawisz ENTER powoduje, oczywiście, powrót karetki i przejście do następnej linii.

Druga grupa funkcji to te rzadziej potrzebne, dostępne poprzez „podwójny" SHIFT.

F i **G** powodują szybkie (22 linie na jedno naciśnięcie) przesuwanie tekstu na ekranie.

P, L i **K** związane są z obsługą drukarki ZX-PRINTER. Służą do wstawiania do tekstu znaczników dużego druku i uruchamiania procesu wydruku.

Bardzo użyteczną funkcją jest R. Umożliwia wyszukanie bądź wyszukanie i zamianę dowolnego tekstu. Niestety, nie działają one zbyt szybko i w przypadku podstawiania trzeba czekać czasem nawet ponad minutę. Ale komputer nie pozwala użytkownikowi na nudę. Cały czas na ekranie widać jak dostawia wybrany przez nas ciąg znaków, porządkując tekst po każdym dostawieniu — aby utrzymać marginesy.

Klawisz **X** służy do kasowania całego pliku. Jest on zabezpieczony przed przypadkowym użyciem.

E włącza i wyłącza rozciąganie tekstu do prawego marginesu w czasie pisania.

J powoduje rozciągnięcie bieżącej linii na całą szerokość ekranu.

H powoduje usunięcie skutków działania rozkazu **J**.

Klawisze **A, S** i **D** służą do ustalania lewego i prawego tabulatora tekstu, ewentualnie kasowania tabulatorów w ogóle.

B, V, N i **M** służą do operowania całymi blokami tekstu.

Autor może zaznaczyć początek i koniec pewnego bloku, po czym skopiować bądź przenieść ten blok w inne miejsce w tekście. Komputer, oczywiście pilnuje estetyki tekstu.

Program pozostawia w pamięci miejsce na tekst o maksymalnej długości 320 linii po 64 znaki. Odpowiada to mniej więcej jedenastu stronom maszynopisu.

Dużą zaletą programu jest fakt, iż współpracuje on z większością popularnych przystawek, takich jak KEMPSTON lub LPRINT III. Gorąco polecam ten program ludziom, którzy gdy pomyślą o przepisywaniu na maszynie, wolą w ogóle nie pisać. Jestem jednym z nich i gdyby nie program TASWORD II, to artykuł ten nigdy by nie powstał.

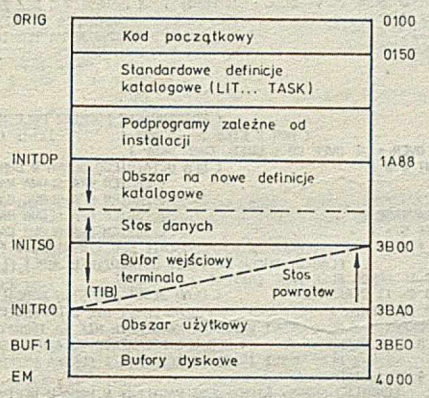
PIOTR PARLEWICZ
Warszawa

[1] Artykuł został nadesłany w takiej postaci, jak przedstawiliśmy w niniejszym akapicie (efekt działania programu TASWORD II); Nie przedstawiamy całego artykułu w tej formie, gdyż brak liter specyficznych dla polskiego alfabetu utrudniłby lekturę — przyp. red.

Gdy dysponujemy pracującą wersją FORTHA, sporządzenie kopii nawet dla nieco innego systemu nie jest zadaniem trudnym. Niemniej — mimo dużego zainteresowania tym językiem — krajowe źródła FORTHA można policzyć na palcach jednej ręki. Opis implementacji „od początku", który zamieszczamy w mikroKLANIE, mimo że poświęcamy mu miejsce w aż trzech kolejnych numerach, jest bardzo lapidarny. Zachęcamy więc Czytelników do samodzielnych prób. W przypadku porażki, listy z konkretnymi pytaniami przekażemy autorowi tekstu, który zainstalował Fig-FORTHA w systemie operacyjnym CP/M.

Jak zainstalować FORTH (2)

Na rysunku 1 przedstawiono mapę pamięci programu Fig-FORTH. Wartości symboli ORIG, INITSO, INITRO i BUF1 (używanych w programie źródłowym) decydują o konfiguracji pamięci. Przedstawione przyporządkowanie **ORIG = 0100H** pozwala na korzystanie z programu Fig-FORTH pod kontrolą systemu operacyjnego CP/M.



Rys. 1. Mapa pamięci programu fig-FORTH

Kod początkowy (tab. 1), oprócz interpretera wewnętrznego NEXT, zawiera program DEBUGGER (praca krokowa) oraz parametry i wskaźniki używane w programie. Poprzez zmianę ich wartości można modyfikować konfigurację systemu w trakcie inicjacji. Może to nastąpić bezpośrednio po włączeniu zasilania (wejście COLD) lub w środku sesji (wejście WARM).

Wykonując słowo VLIST, można uzyskać listę nazw wszystkich standardowych definicji katalogowych (wydruk 1).

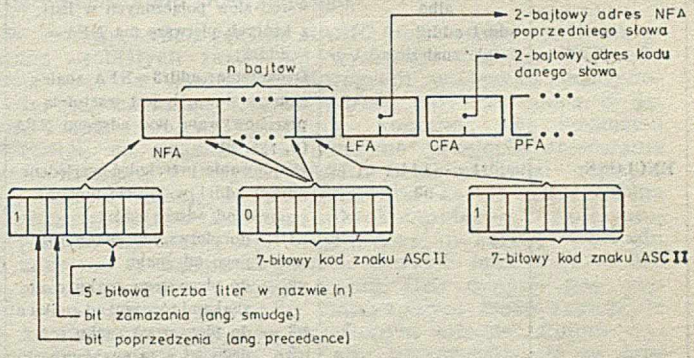
```
VLIST
TASK  .CPU  TRIAD  INDEX  LIST  BYE  VLIST  U.  ?  .  D.
.R  D.R  $S  $  SIGN  #>  <#  SPACES  WHILE  ELSE  IF
REPEAT  AGAIN  END  UNTIL  +LOOP  LOOP  DO  THEN  ENDIF
BEGIN  BACK  FORGET  '  -->  LOAD  FLUSH  R/W  SEC-WRITE
SEC-READ  TASCALC  SET-DRIVE  SET-ID  BLOCK  BUFFER  DR1
DR0  EMPTY-BUFFERS  UPDATE  +BUF  DISK-ERROR  DENSITY  #BUFF
SEC/BLK  PREV  USE  TRACK  SEC  DRIVE  P!  PG  MESSAGE
.LINE  (LINE)  M/MOD  */  */MOD  MOD  /  /MOD  M  R/  R#
MAX  MIN  DABS  ABS  D+-  +-  S->D  COLD  WARM  ABORT
QUIT  (  DEFINITIONS  FORTH  VOCABULARY  IMMEDIATE  INTERPRET
?STACK  DLITERAL  LITERAL  [COMPILE]  CREATE  ID.  ERROR
(ABORT)  -FIND  NUMBER  (NUMBER)  WORD  PAD  HOLD  BLANKS
ERASE  FILL  QUERY  EXPECT  ."  (.")  -TRAILING  TYPE
COUNT  DOES>  <BUILDS  ;CODE  (;CODE)  DECIMAL  HEX  SMUDGE
]  [  COMPILE  ?LOADING  ?CSP  ?PAIRS  ?EXEC  ?COMP  ?ERROR
!CSP  PFA  NFA  CFA  LFA  LATEST  TRAVERSE  -DUP  SPACE
ROT  >  U<  <  =  -  C,  ,  ALLOT  HERE  2+  1+  HLD
R#  CSP  FLD  DPL  BASE  STATE  CURRENT  CONTEXT  OFFSET
SCR  OUT  IN  BLK  VOC-LINK  DP  FENCE  WARNING  WIDTH
TIB  R0  S0  +ORIGIN  B/SCR  B/BUF  LIMIT  FIRST  C/L  BL
3  2  1  0  USER  VARIABLE  CONSTANT  NOOP  -1  1  2!
C!  !  2@  C@  @  TOGGLE  +!  2DUP  DUP  SWAP  DROP  OVER
MINUS  MINUS  D+  +  0<  0=  R  R>  >R  LEAVE  #S  R#!
RP@  SP!  SP@  XOR  OR  AND  U/  U*  CMOVE  CR  ?TERMINAL
KEY  EMIT  ENCLOSE  (FIND)  DIGIT  I  (DO)  (+LOOP)  (LOOP)
0BRANCH  BRANCH  EXECUTE  LIT  0K
```

Wydruk 1

Tabela 1. Kod początkowy

Adres względny	Adres bezwzględny	Opis
ORIG+00H	0100H	wejście COLD
ORIG+04H	0104H	wejście WARM
ORIG+08H	0108H	parametry wejścia WARM
ORIG+12H	0112H	parametry wejścia COLD
ORIG+26H	0126H	adres początku obszaru użytkowego
ORIG+28H	0128H	wskaźnik stosu powrotów
ORIG+2AH	012AH	debugger
ORIG+45H	0145H	interpreter wewnętrzny NEXT

Na rysunku 2 przedstawiono strukturę definicji katalogowej. Składa się ona z czterech pól: pola nazwy (ang. name field), pola łącznika (ang. link field), pola kodu (ang. code field) i pola parametrów (ang. parameter field). Symbole NFA, LFA, CFA, i PFA oznaczają adresy tych pól.



Rys. 2. Struktura definicji katalogowej

Tabela 2 przedstawia podział definicji katalogowych na grupy. Później definiowane słowa składają się na ogół z sekwencji wywołań słów zdefiniowanych wcześniej. Dlatego też procedury napisane w kodzie maszynowym występują na samym początku. Tabela 3 zawiera opis pomocniczych słów w kodzie maszynowym, nie omówionych dotąd w INFORMATYCE. Oprócz odwołań „w tył", zdarzają się także odwołania „w przód" do słów występujących później (ang. forward definitions). W dalszej kolejności występują słowa-kompilatory. Za ich pomocą można definiować nowe słowa (m.in. stałe, zmienne użytkowe i procedury w kodzie nizanym).

Tabela 2. Standardowe definicje katalogowe

Nazwy słów	Opis
LIT ,,,2! : ; CONSTANT VARIABLE USER	procedury w kodzie maszynowym kompilatory
0 ... B/SCR	stałe
+ORIGIN	specjalna procedura
SO ... HLD	zmienne użytkowe
1+ ... P!	procedury w kodzie nizanym
DRIVE ... -->	stałe, zmienne i procedury do współpracy z dyskami
, FORGET BACK BEGIN ... WHILE SPACES ... TASK	specjalne procedury instrukcje strukturalne

Stałe programu (wydruk 2), zdefiniowane przy użyciu kompilatora CONSTANT, umożliwiają odwołania przez nazwę do najczęściej używanych liczb (zamiast pośredni-

ctwa słowa LIT), skracając program w kodzie nizanym. Definiują też pewne wartości zależne od implementacji, dając możliwość łatwej ich zmiany.

Tabela 3. Niektóre procedury pomocnicze w kodzie maszynowym

Nazwa słowa	Stos	Opis
EXECUTE	addr→	wykonanie słowa, w którym CFA = addr
DIGIT	c n1→n2 1 (konwersja możliwa) albo c n1→ 0 (konwersja niemożliwa)	zamiana znaku ASCII c zgodnie z podstawą n1 na liczbę n2
(FIND)	addr1 addr2→ addr3 b 1 (znaleziono) albo addr1 addr2→0 (nie znaleziono)	poszukiwanie słowa identycznego z tekstem umieszczonym od adresu addr1 (zakończonego spacją) wśród słów połączonych w listę, z których pierwsze ma NFA= =addr2; Oznaczenia; addr3=NFA znalezionego słowa, b jest wartością przechowywaną pod adresem NFA (p. rys. 2)
ENCLOSE	addr1 c→addr1 n1 n2 n3	znajdowanie przesunięć względem adresu addr1 w tekście umieszczonym od adresu addr1: n1 — do pierwszego znaku różnego od znaku c n2 — do pierwszego znaku c za znakami różnymi od znaku c n3 — do pierwszego znaku różnego od c za znakiem o przesunięciu n2
CMOVE	addr1 addr2 n→	przepisanie n bajtów z obszaru (addr1; addr1+n) w obszar (addr2; addr2+n)
TOGGLE	addr b→	zmiana wartości tych bitów w bajcie umieszczonym pod adresem addr, którym w masce b odpowiada „1"

```
HEX
  00 CONSTANT 0
  01 CONSTANT 1
  02 CONSTANT 2
  03 CONSTANT 3
  20 CONSTANT BL    ( SPACJA)
  50 CONSTANT C/L   ( ILOSC ZNAKOW W LINII = 80D)
3BE0 CONSTANT FIRST ( ADRES PIERWSZEGO BUFORA DYSKOWEGO)
4000 CONSTANT LIMIT ( ADRES OSTATNIEGO BAJTU PAMIECI+1)
  80 CONSTANT B/BUF ( ILOSC BAJTOW DANYCH W BUFORZE DYSKOWYM-128D)
   8 CONSTANT B/SCR ( ILOSC BUFOROW W KADRZE = 8D)
```

Wydruk 2

Zmienne programu (wydruk 3) pamiętane są w obszarze użytkowym (rys. 1.). Podczas definiowania, z wykorzystaniem słowa USER, podaje się adres względny w ramach tego obszaru. Przy inicjacji przez wejście COLD niektórym z tych zmiennych nadawane są wartości początkowe.

Do definiowania słów w kodzie nizanym służy para kompilatorów: „:" i „;". W kodzie nizanym zapisana jest większość słów FORTHA, w tym także interpreter zewnętrzny. Omówienie implementacji ciekawszych słów FORTHA odłożymy do trzeciej części artykułu, poprzestając teraz na przytoczeniu tekstu źródłowego z minimalnymi komentarzami (wydruk 4).

Tym, którzy mają dostęp do instalacji fig-FORTHA proponuję sprawdzenie przedstawionych tu definicji — poprzez wprowadzenie i wykonanie (z wyjątkiem **<BUILDS** i **DOES>**) programów z wydruków 2, 3 i 4. Oczywiście, zainstalowana wersja FORTHA zawiera już te słowa. Wprowadzając je po raz drugi, uzyskujemy coś w rodzaju niepełnej kopii systemu. Wejście do nowej kopii następuje przez wykonanie słowa QUIT (zgłoszenie następuje wtedy komunikatem „KO" zamiast „OK"). Powrót do „starej" wersji następuje po wystąpieniu jakiegokolwiek błędu. Nowa kopia jest niepełna, m. in. dlatego, że nie ma nowych procedur w kodzie maszynowym. Można by — korzystając z asemblera fig-FORTHA — zdefiniować powtórnie także i te słowa. W ten sposób możliwe jest generowanie nowych wersji FORTHA działających przy innej konfiguracji sprzętowej, a nawet na innym procesorze (przy użyciu odpowiedniego programu asemblera). Nowa wersja mo-

```
06 USER S0          ( SZCZYT PUSTEGO STOSU DANYCH)
08 USER R0          ( SZCZYT PUSTEGO STOSU POWROTOW)
0A USER TIB         ( BUFOR WEJSCIOWY TERMINALA)
0C USER WIDTH       ( MAKSYMALNA ILOSC ZNAKOW W NAZWIE)
0E USER WARNING     ( STERUJE REAKCJA NA BLEDY)
10 USER FENCE       ( GRANICA ZAPOMINANIA DEFINICJI PRZY UZYCIU FORGET)
12 USER DP          ( WSKAZNIK SLOWNIKA)
14 USER VOC-LINK    ( WSKAZNIK DO PFA+6 OSTATNIO ZDEFINIOWANEGO SLOWNIKA)
16 USER BLK         ( NUMER INTERPRETOWANEGO BLOKU LUB 0 GDY TIB)
18 USER IN          ( WSKAZNIK ZNAKOW BUFORA WEJSCIOWEGO)
1A USER OUT         ( POZYCJA KURSORA NA EKRANIE)
1C USER SCR         ( NUMER EDYTOWANEGO KADRU)
1E USER OFFSET      ( 0 -GDY WYBR. DRIVE 0, IL. SEK. NA DYSKU - GDY DR.1)
20 USER CONTEXT     ( ADRES PFA+4 SLOWNIKA PRZESZUKIWANEGO JAKO )
                    (          PIERWSZY PRZY KOMPILOWANIU SLOWA )
22 USER CURRENT     ( ADR. PFA+4 SLOWN. DO KT. DOLACZA SIE DEF.)
24 USER STATE       ( 00 - STAN EGZEKUCJI, C0 - KOMPILACJI)
26 USER BASE        ( PODSTAWA SYSTEMU LICZBOWEGO)
28 USER DPL         ( POZYCJA KROPKI DZIESIETNEJ)
2A USER FLD         ( SZEROKOSC POLA WYSWIETLANEJ LICZBY )
2C USER CSP         ( MIEJSCE NA ZAPAMIETYWANIE WSKAZNIKA STOSU)
2E USER R#          ( POZYCJA KURSORA EDYTORA)
30 USER HLD         ( WSKAZNIK OSTATNIEGO ZNAKU W BUFORZE PAD)
```

Wydruk 3

```
: 1+ 1 + ;                                  ( ZWIEKSZ O 1 WARTOSC NA STOSIE)
: 2+ 2 + ;                                  ( ZWIEKSZ O 2 WARTOSC NA STOSIE)
: HERE DP @ ;                       ( PODAJE NASTEPNY WOLNY ADRES W SLOWNIKU)
: ALLOT DP +! ;                             ( PRZESUN WSKAZNIK SLOWNIKA)
: , HERE ! 2 ALLOT ;                ( WPISZ DO SLOWNIKA WARTOSC ZE STOSU)
: C, HERE C! 1 ALLOT ;                      ( WPISZ DO SLOWNIKA BAJT ZE STOSU)
: - MINUS + ;                       ( ODEJMIJ WARTOSC OSTATNIA OD PRZEDOSTATNIEJ)
: = - 0= ;                          ( WARTOSC LOGICZNA 'PRZEDOSTATNIA=OSTATNIA')
: < - 0< ;                          ( WARTOSC LOGICZNA 'PRZEDOSTATNIA<OSTATNIA')
: U< 2DUP XOR 0< IF DROP 0< 0= ELSE - 0< ENDIF ;  ( NIEJEDNOSC BEZ ZNAKU)
: ROT >R SWAP R> SWAP ;             ( ROTUJ PRZEDPRZEDOSTATNIA NA WIERZCH)
: SPACE BL EMIT ;                           ( WYSWIETL SPACJE NA TERMINALU)
: -DUP DUP IF DUP ENDIF ;                   ( DUPLIKUJ WARTOSC NIEZEROWA)



: TRAVERSE                                  ( PRZESUN SIE POPRZEZ POLE NAZWY)
                                    ( ADR-2 KIER-1 --- ADR-1     , -1(--, +1--))
   SWAP BEGIN OVER + 7F OVER C@ < UNTIL SWAP DROP ;
: LATEST CURRENT @ @ ;                      ( NFA OSTATNIEGO SLOWA W CURRENT)
: LFA 4 - ;                                 ( ZAMIEN PFA SLOWA NA LFA)
: CFA 2 - ;                                 ( ZAMIEN PFA SLOWA NA CFA)
: NFA 5 - -1 TRAVERSE ;                     ( ZAMIEN PFA SLOWA NA NFA)
: PFA 1 TRAVERSE 5 + ;                      ( ZAMIEN NFA SLOWA NA PFA)
: !CSP SP@ CSP ! ;                  ( ZAPAMIETAJ POLOZENIE SZCZYTU STOSU W 'CSP')
: ?ERROR                    ( FLAG-2, NUMER BLEDU-1, OSTRZEZENIE JESLI FLAG-1)
        SWAP IF ERROR ELSE DROP ENDIF ;
: ?COMP STATE @ 0= 11 ?ERROR ;              ( BLAD JESLI NIE STAN KOMPILACJI)
: ?EXEC STATE @ 12 ?ERROR ;                 ( BLAD JESLI NIE STAN EGZEKUCJI)
: ?PAIRS - 13 ?ERROR ;  ( BLAD JESLI WARTOSCI NA STOSIE NIE SA IDENTYCZNE)
: ?CSP SP@ CSP @ - 14 ?ERROR ;      ( SPRAWDZ POLOZENIE WSKAZNIKA STOSU)
: ?LOADING                  ( BLAD JESLI INTERPRETACJA NIE Z BUFORA DYSKOWEGO)
        BLK @ 0= 16 ?ERROR ;
: COMPILE       ( KOMPILUJ ADRES, KTORY JEST NASTEPNY W RAMACH DEFINICJI)
        ?COMP R> DUP 2+ >R @ , ;
: [ 0 STATE ! ; IMMEDIATE                   ( ZATRZYMAJ KOMPILACJE)
: ] C0 STATE ! ;                            ( WEJDZ W STAN KOMPILACJI)
: SMUDGE LATEST 20 TOGGLE ;  ( ZMIEN BIT SMUDGE W OSTATNIM NAGLOWKU SLOWA)
: HEX 10 BASE ! ;                           ( PRZYJMIJ PODSTAWE 16D DLA WE-WY)
: DECIMAL 0A BASE ! ;                       ( PRZYJMIJ PODSTAWE 10D DLA WE-WY)
: (;CODE)   ( ZAPISZ POLE KODU WSKAZNIKIEM DO DALSZEGO OBSZARU DEFINICJI)
        R> LATEST PFA CFA ! ;
: ;CODE ?CSP COMPILE (;CODE)                ( KONCZ NOWE SLOWO DEFINIUJACE)
        [COMPILE] [ SMUDGE NOOP ( ASSEMBLER ) ; IMMEDIATE
: (BUILDS 0 CONSTANT ;                      ( UTWORZ NAGLOWEK SLOWA)
: DOES) ( WPISZ AD. WYS. POZIOMU DO PFA, DO CFA - AD. KODU 'DOES)'-DODOE)
    R> LATEST PFA ! ;CODE ( ... KOD PROCEDURY DOCOL ...) C;
: COUNT DUP 1+ SWAP C@ ;            ( USTAL ADRES-2, LICZBE ZNAKOW-1 TEKSTU)
: TYPE                      ( WYPROWADZ TEKST OD ADRESU-2, LICZBA ZNAKOW-1)
      -DUP IF OVER + SWAP DO I C@ EMIT LOOP ELSE DROP ENDIF ;
: -TRAILING             ( USTAL LICZBE ZNAKOW PO ODRZUCENIU KONCOWYCH SPACJI)
      DUP 0 DO OVER OVER + 1 - C@ BL - IF LEAVE ELSE 1 - ENDIF LOOP ;
: (.")          ( WYPROWADZ UMIESZCZONY DALEJ TEKST, DOSTOSUJ SLAD POWROTU)
    R COUNT DUP 1+ R> + >R TYPE ;
: ."    22 STATE @          ( KOMPILUJ LUB WYPROWADZ TEKST POMIEDZY ." A ")
    IF COMPILE (.") WORD HERE C@ 1+ ALLOT
    ELSE WORD HERE COUNT TYPE ENDIF ;       IMMEDIATE
: EXPECT                        ( ADRES BUFORA NA TEKST-2, LIMIT ZNAKOW -1)
    OVER + OVER
      DO KEY DUP 0E +ORIGIN @ =
        IF DROP DUP I = DUP R> 2 - + >R IF 07 ELSE 08 ENDIF
        ELSE DUP 0D = IF LEAVE DROP BL 0 ELSE DUP ENDIF I C! 0 I 1+ !
        ENDIF EMIT
      LOOP DROP ;
```

```
: QUERY TIB @ 50 EXPECT 0 IN ! ;  ( POBIERZ TEKST Z KONSOLI DO BUFORA TIB)
8081 ( ZAWART. POLA NAZWY ZLOZ. Z JEDNEGO ZNAKU '00' ) HERE ( ADR. POLA N.)
: X      BLK @  (  NULL KONCZACY W BUFORZE TEXT - POWROT NA WYZSZY POZIOM)
 IF ( DYSK.) 1 BLK +! 0 IN ! BLK @ B/SCR 1 - AND 0= IF ?EXEC R) DROP ENDIF
 ELSE ( TIB) R) DROP ENDIF ; ! ( ZAMIEN NAZWE X NA '00' ) IMMEDIATE
: FILL                          ( WYPELNIJ PAMIEC OD ADR-3, ILOSCIA-2 BAJTOW-1)
    SWAP )R OVER C! DUP 1( R) 1 - CMOVE ;
: ERASE 0 FILL ;                ( WYPELNIJ PAMIEC ZERAMI OD ADR-2, ILOSC -1)
: BLANKS BL FILL ;              ( WYPELNIJ PAMIEC SPACJAMI OD ADR-2, ILOSC-1)
: HOLD -1 HLD +! HLD @ C! ;                  ( SCHOWAJ ZNAK W BUFORZE PAD)
: PAD HERE 44 + ;                ( PAD ZNAJDUJE SIE 68D BAJTOW POWYZEJ HERE)
: WORD  ( WPROW. TEKST WEJSC.  AZ DO OGRANICZNIKA, UMIESC TEKST W BUFORZE)
     BLK @ IF BLK @ BLOCK ELSE TIB @ ENDIF          ( I PRZESLIJ DO 'HERE')
     IN @ + SWAP                                  ( ADRES-2, OGRANICZNIK-1)
     ENCLOSE          ( ADRES-4, POCZATEK-3, KONIEC-2, CALKOWITA-ILOSC-1)
     HERE 22 BLANKS                  ( WYCZYSC BUFOR WPISUJAC 34D SPACJE)
     IN +!                             ( PRZESKOCZ PRZEZ LANCUCH W TIB)
     OVER - )R                                      ( CHRON ILOSC ZNAKOW)
     R HERE C!                        ( PRZED TEKSTEM ZAPISZ ILOSC ZNAKOW)
     + HERE 1+
     R) CMOVE ;           ( PRZEPISZ LANCUCH ZNAKOW Z BUFORA TIB DO HERE+1)
: (NUMBER)                                      ( D1 ADR1 ---> D2 ADR2)
                ( DODAJ DO PODWOJNEJ LICZBY D1 LICZBE ZAPISANA W POSTACI)
   ( ZNAKOW ASCII POD ADR1, WYNIK PODWOJNA LICZBA D2, ADR2=ADR1+IL.ZNAKOW)
 BEGIN 1+ DUP )R C@ BASE @ DIGIT
 WHILE SWAP BASE @ UM* DROP ROT BASE @ UM* D+ DPL @ 1+ IF 1 DPL +! ENDIF R)
 REPEAT R) ;
: NUMBER ( ZAMIEN ZN. ASCII POD ADR NA LICZBE PODWOJNA D; JESLI NAPOTKANO)
                                          ( . 10 POZYCJE KROPKI ZAPAMIETAJ W DPL )
 0 0 ROT DUP 1+ C@ 2D = DUP )R + -1
 BEGIN DPL ! (NUMBER) DUP C@ BL -
  WHILE DUP C@ 2E - 0 ?ERROR 0 REPEAT
  DROP R) IF DMINUS ENDIF ;
: -FIND                                         ( ZNAJDZ SLOWO W SLOWNIKU)
                    ( PO WYKONANIU: PFA 3, BAJT DLUGOSCI-2,1 ALBO TYLKO 0)
   BL WORD HERE CONTEXT @ @ (FIND)
   DUP 0= IF DROP HERE LATEST (FIND) ENDIF ;
: (ABORT) ABORT ;                    ( ABORT ZMIENIANY PRZEZ UZYTKOWNIKA)
: ERROR (  W ZALEZNOSCI OD WARNING:)
     ( -1=ABORT, 0=BRAK DYSKOW-NUMER BLEDU, 1= NA DYSKI-KOMUN. O BLEDZIE)
  WARNING @ 0<
  IF (ABORT) ENDIF HERE COUNT TYPE ." ? "
  MESSAGE SP! BLK @ -DUP IF IN @ SWAP ENDIF QUIT ;
: ID.                          ( WYPROWADZ NAZWE NA PODSTAWIE ADRESU NFA)
    PAD 020 5F FILL DUP PFA LFA OVER
    PAD SWAP CMOVE PAD COUNT 01F AND TYPE SPACE ;



: CREATE  ( UTWORZ NAGLOWEK Z USTAWIONYM BITEM SMUDGE, WYSLIJ OSTRZEZENIE)
                        ( JESLI NOWA NAZWA JUZ WCZESNIEJ BYLA ZDEFINIOWANA)
    TIB HERE 0A0 + U< 2 ?ERROR -FIND
    IF ( OSTRZEZENIE) DROP NFA ID. 4 MESSAGE SPACE ENDIF
    HERE DUP C@ WIDTH @ MIN 1+ ALLOT
    DUP 0A0 TOGGLE HERE 1 - 80 TOGGLE    ( USTAW BITY BAJTOW GRANICZNYCH)
    LATEST , CURRENT @ !
    HERE 2+ , ;
: [COMPILE]                        ( WYMUSZA KOMPILACJE NATYCHMIASTOWEGO SLOWA)
    -FIND 0= 0 ?ERROR DROP CFA , ; IMMEDIATE
: LITERAL          ( JESLI STAN KOMPILACJI, KOMPILUJE LIT I LICZBE ZE STOSU)
    STATE @ IF COMPILE LIT , ENDIF ; IMMEDIATE
: DLITERAL     ( JESLI STAN KOMPILACJI, KOMPILUJE PODWOJNA LICZBE ZE STOSU)
    STATE @ IF SWAP [COMPILE] LITERAL [COMPILE] LITERAL ENDIF ; IMMEDIATE
: ?STACK  ( SPRAWDZ, CZY WSKAZNIK STOSU NIE WYSZEDL POZA DOPUSZCZ. ZAKRES)
SP@ S0 @ SWAP U< 1 ?ERROR SP@ HERE 80 + U< 7 ?ERROR ;
: INTERPRET     ( INTERPRETUJ LUB KOMPILUJ SLOWA ZRODL. TEKSTU WEJSCIOWEGO)
    BEGIN -FIND
        IF ( ZNALEZIONO) STATE @ (
            IF ( KOMPILUJ) , ELSE ( WYKONAJ) EXECUTE ENDIF ?STACK
        ELSE ( NIE ZNALEZIONO) HERE NUMBER DPL @ 1+
            IF ( BYLA KROPKA) [COMPILE] DLITERAL
            ELSE ( NIE BYLO KROPKI) DROP [COMPILE] LITERAL
            ENDIF ?STACK
        ENDIF
    AGAIN ;
: IMMEDIATE        ( ZMIEN BIT POPRZEDZANIA (PRECEDENCE) OSTATNIEGO SLOWA)
    LATEST 40 TOGGLE ;
: VOCABULARY                       ( UTWORZ SLOWO BEDACE NOWYM SLOWNIKIEM)
    <BUILDS A081 , CURRENT @ CFA , HERE VOC-LINK @ , VOC-LINK !
    DOES> 2+ CONTEXT ! ;
VOCABULARY FORTH IMMEDIATE                                ( SLOWNIK FORTH )
: DEFINITIONS                    ( ZE SLOWNIKA CONTEXT UCZYN TAKZE CURRENT)
    CONTEXT @ CURRENT ! ;
: (                ( PRZESKOCZ TEXT WEJSCIOWY AZ DO PRAWEGO NAWIASU OKACZNIE)
    29 ( PRAWY NAWIAS) WORD ; IMMEDIATE
: QUIT                                    ( RESTART, INTERPRETUJ Z TERMINALA)
   0 BLK ! [COMPILE] [
   BEGIN RP! CR QUERY INTERPRET STATE @ 0= IF ." OK" ENDIF AGAIN ;
: ABORT SP! HEX ?STACK CR ." FIG FORTH V1.1"          ( "CORACY" RESTART)
    [COMPILE] FORTH DEFINITIONS QUIT ;
```

Wydruk 4

że być pozbawiona pola nazwy i pola łącznika w definicjach katalogowych, gdyż przy realizacji konkretnego zadania nie są one potrzebne. Daje to możliwość uzyskania krótkiego kodu (często krótszego od napisanego w asemblerze), przy jednoczesnym szybkim pisaniu i uruchamianiu programu. Proces generowania nowych wersji FORTHA nazywany jest **metakompilacją.**

ZBIGNIEW POJMAŃSKI

Stosowanie kodu maszynowego w komputerach osobistych

Popularne komputery osobiste zamiast programu MONITOR wyposażone są zazwyczaj w interpreter języka BASIC umieszczony w pamięci stałej. Ma to swoje zalety, jeśli przyjąć, że użytkownikiem będzie tzw. odbiorca masowy, który rzadko potrzebuje bardziej wyrafinowanych narzędzi. Niestety, mikroprocesory 8-bitowe mają zbyt małą moc obliczeniową, aby można było nawet w popularnych zastosowaniach ograniczyć się do stosowania wyłącznie języków wysokiego poziomu. Zachodzi więc konieczność wykorzystania, przynajmniej w niektórych podprogramach, kodu maszynowego.

W interpreterach języka BASIC stosowanych w komputerach osobistych zazwyczaj występuje funkcja **USR adr,** która pozwala wywołać podprogram w języku maszynowym, rozpoczynający się od adresu **adr.** Procedura w języku maszynowym powinna kończyć się rozkazem **RETURN,** co pozwala na ponowne przejęcie sterowania przez interpreter języka BASIC. Jeśli zachodzi potrzeba przekazania danej, to powinna ona zostać umieszczona w parze rejestrów BC (dla mikrokomputerów wykorzystujących Z80 lub 8080). Jeżeli więc napiszemy **LET A=USR adr,** to zmienna A będzie zawierała wartość jaka była w BC w chwili powrotu z podprogramu. Gdy danych jest więcej — procedura w języku maszynowym może zapisać je w komórkach pamięci o określonym adresie. Potem wystarczy wykonać **PEEK adr,** aby wykorzystać je w programie napisanym w języku BASIC. Analogicznie przez **POKE** można przekazać dane dla procedury w języku maszynowym. Gdy nie zachodzi potrzeba przekazania danych z podprogramu, najczęściej stosuje się zapis: **RANDOMIZE USR adr.**

Pozostaje jeszcze kwestia zapisu programów w języku maszynowym. Tradycyjną i zarazem najwygodniejszą metodą jest wykorzystanie do tego celu pogramów EDITOR i ASSEMBLER [1]. Pierwszy pozwala na wprowadzenie programu w postaci symbolicznej. Powstaje w ten sposób tzw. wersja źródłowa programu (ang. source), która jest znacznie bardziej przejrzysta dla człowieka niż zestaw cyferek. Za pomocą programu ASSEMBLER z wersji źródłowej naszego programu otrzymujemy tzw. wersję wynikową, czyli kod, który po zapisaniu do pamięci może być interpretowany przez procesor.

W niektórych mikrokomputerach programy EDITOR i ASSEMBLER zawarte są w pamięci stałej. W przypadku komputerów osobistych trzeba je zazwyczaj ładować do pamięci z taśmy magnetofonowej lub z dysku.

Co zrobić, jeśli nie dysponujemy programami EDITOR i ASSEMBLER? Praktycznie nie warto zabierać się za pisanie własnych programów w języku maszynowym. Wprawdzie można pobawić się wyszukując w książce kod dla każdej instrukcji, ale ma to sens tylko w przypadku bardzo krótkich procedur.

Tylko niektóre komputery osobiste mają możliwość zapamiętania programu w kodzie maszynowym poprzez zapisanie go na taśmie (lub dysku). Przykładowo — dla ZX81 trzeba zastosować pewien trik; kod maszynowy umieszcza się w pierwszej linii programu za instrukcją REM — w ten sposób nietrudno określić adres początku, a kod razem z resztą programu można bez problemu zapisać na kasecie magnetofonowej.

---

[1]) Słowo ASSEMBLER używane jest również dla określenia języka maszynowego

Przy publikowaniu programów dla mikrokomputerów bardzo często fragmenty pisane w języku maszynowym zamieszcza się pod postacią tablicy kodów. Drukowanie wersji źródłowej zajmuje znacznie więcej miejsca i dodatkowo ogranicza krąg odbiorców — do tych, którzy dysponują programem ASSEMBLER. Wprowadzanie (sprawdzonych już przez kogoś) kodów jest zapewne bardziej żmudne niż wprowadzanie wersji źródłowej — sumarycznie zajmuje jednak mniej czasu. Kody zazwyczaj drukuje się w liniach zawierających: adres, 16 cyfr szesnastkowych i sumę kontrolną. Suma kontrolna pozwala na wykrycie błędów popełnionych przez wprowadzającego. Dla ułatwienia wprowadzania stosuje się specjalne programy ładujące kody szesnastkowe we wskazane miejsce pamięci (ang. hex loader).

Począwszy od tego wydania mikroKLANU publikować będziemy co jakiś czas programy dla użytkowników ZX81. Jeśli będą one zawierać fragmenty napisane w języku maszynowym, proponujemy do ich wprowadzania wykorzystywać następujący program ładujący:

```
6900 REM ZAPIS
7000 CLS
7010 PRINT "WPISYWANIE CZY SPRAWDZANIE? W/S"
7020 INPUT M$
7030 PRINT "ADRES POCZATKU"
7040 INPUT S
7050 PRINT "ADRES KONCA"
7060 INPUT F
7070 IF M$="S" THEN GOTO 8000
7080 FOR N=S TO F STEP 8
7090 LET T=0
7100 PRINT N;"—";
7110 INPUT A$
7120 PRINT A$;"=";
7130 INPUT TOT
7140 PRINT TOT
7145 LET Z=0
7150 FOR K=1 TO LEN A$ STEP 2
7160 LET C=(CODE A$(K)—28)*16+CODE A$(K+1)—28
7170 LET T=T+C
7180 POKE N+Z,C
7190 LET Z=Z+1
7200 NEXT K
7210 IF TOT=T THEN GOTO 7240
7220 SCROLL
7225 PRINT "BLAD — JESZCZE RAZ"
7230 GOTO 7090
7240 NEXT N
7250 STOP
7900 REM SPRAWDZANIE
8000 FOR N=S TO F STEP 8
8005 SCROLL
8010 PRINT N;"—";
8020 FOR B=0 TO 7
8030 LET C=PEEK (N+B)
8040 LET F$=CHR$ (INT (C/16+28))
8050 LET S$=CHR$((C—((CODE F$—28)*16))+28
8055 PRINT F$;S$;
8060 NEXT B
8070 PRINT
8080 NEXT N
```

Zanim użyty zostanie program ładujący należy napisać: 1 REM i tyle znaków, ile bajtów liczy program. Znaki te mogą być najzupełniej dowolne — ważna jest ich liczba.

Zaprezentowany program ładujący można również wykorzystać i dla innych mikrokomputerów (np. ZX SPECTRUM). Należy go jednak odpowiednio przystosować (ZX81 nie wykorzystuje kodów ASCII).

AJP

prowadzi:
Andrzej J. Piotrowski
tel. dom. 48-22-85
tel. służb. 43-66-11 w. 98

Wielu czytelników mikroKLANU stawia dopiero swoje pierwsze mikroinformatyczne kroki. Nie jesteśmy w stanie przy obecnej objętości mikroKLANU zamieszczać materiałów na poziomie zupełnie elementarnym. Wspomagając tych co stawiają krok drugi lub trzeci publikujemy między innymi proste programy w języku BASIC, wychodząc z założenia, że najlepiej uczyć się na przykładach.

Zamiana liczb arabskich na rzymskie

```
 5 REM ZAMIANA LICZB ARABSKICH NA RZYMSKIE
 6 REM RZYMIANIE NIE MIELI OZNACZENIA NA
 7 REM LICZBY MNIEJSZE NIZ 1 ORAZ NA
 8 REM WIEKSZE NIZ 4999.
 10 DIM A(13),R(13)
 20 FOR I=1 TO 13
 30 READ A(I)
 40 NEXT I
 50 DATA 1000,900,500,400,100,90,50,40,10,9,5,4,1
 60 PRINT "LICZBA ARABSKA (1-4999)?"
 61 INPUT N
 70 IF N<1 OR N>4999 THEN GOTO 60
 80 PRINT N;"=";
 90 FOR I=1 TO 13
 100 R(I)=0
 110 M=N-A(I)
 120 IF M<0 THEN GOTO 160
 130 R(I)=R(I)+1
 140 N=M
 150 GOTO 110
 160 NEXT I
 165 IF R(1)=0 THEN GOTO 200
 170 FOR I=1 TO R(1)
 180 PRINT "M";
 190 NEXT I
 200 IF R(2)=1 THEN PRINT "CM";
 210 IF R(3)=1 THEN PRINT "D";
 220 IF R(4)=1 THEN PRINT "CD";
 225 IF R(5)=0 THEN GOTO 260
 230 FOR I=1 TO R(5)
 240 PRINT "C";
 250 NEXT I
 260 IF R(6)=1 THEN PRINT "XC";
 270 IF R(7)=1 THEN PRINT "L";
 280 IF R(8)=1 THEN PRINT "XL";
 285 IF R(9)=0 THEN GOTO 320
 290 FOR I=1 TO R(9)
 300 PRINT "X";
 310 NEXT I
 320 IF R(10)=1 THEN PRINT "IX";
 330 IF R(11)=1 THEN PRINT "V";
 340 IF R(12)=1 THEN PRINT "IV";
 345 IF R(13)=0 THEN GOTO 380
 350 FOR I=1 TO R(13)
 360 PRINT "I";
 370 NEXT I
 380 PRINT
 390 PRINT "NASTEPNA LICZBA (0-KONIEC)?"
 391 INPUT N
 400 IF N>0 THEN GOTO 70
 410 END
READY.
    RUN
LICZBA ARABSKA (1-4999)?
? 1984
 1984 =MCMLXXXIV
NASTEPNA LICZBA (0-KONIEC)?
? 2865
 2865 =MMDCCCLXV
NASTEPNA LICZBA (0-KONIEC)?

? 0
READY.
```

J.T.

W mikroKLANIE 6 pod artykułem Władysława Minora nt. mikrokomputera NEPTUN 184 (INFORMATYKA nr 6/84, s. 14) zamieściliśmy następującą adnotację: „Artykuł ten został opublikowany w porozumieniu z producentem i przez niego sfinansowany". Wywołała ona nieporozumienia, autor zaś poczuł się urażony posądzeniem, iż występuje jako rzecznik producenta. Wyjaśniamy więc, że — zgodnie z wcześniejszymi zapowiedziami — staramy się uzyskać od wytwórców sprzętu mikrokomputerowego dotacje finansowe związane z artykułami nt. ich produktów, przy czym „cenzury producenta" nie uznajemy. Pracownia Elektroniki Medycznej, firmująca NEPTUNA, jako pierwsza przyjęła, że wsparcie INFORMATYKI ma szersze uzasadnienie; zgodziła się więc na wykupienie kolumn pisma przeznaczonych na artykuł Władysława Minora. Sam autor w ogóle o tej transakcji nie wiedział — tekst nie został w jej wyniku naruszony. (Red.)